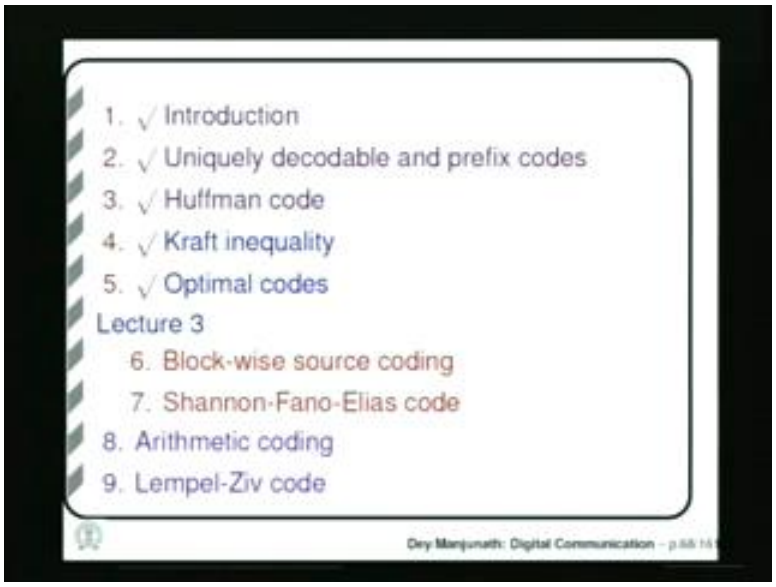
Digital Communication
Prof. Bikash Kumar Dey
Department of Electrical Engineering
Indian Institute of Technology, Bombay

Lecture - 28
Source Coding (Part – 3)

We will continue in this class with source coding. So, this is the third lecture on source coding.

(Refer Slide Time: 01:20)



1. √ Introduction
2. √ Uniquely decodable and prefix codes
3. √ Huffman code
4. √ Kraft inequality
5. √ Optimal codes

Lecture 3

6. Block-wise source coding
7. Shannon-Fano-Elias code
8. Arithmetic coding
9. Lempel-Ziv code

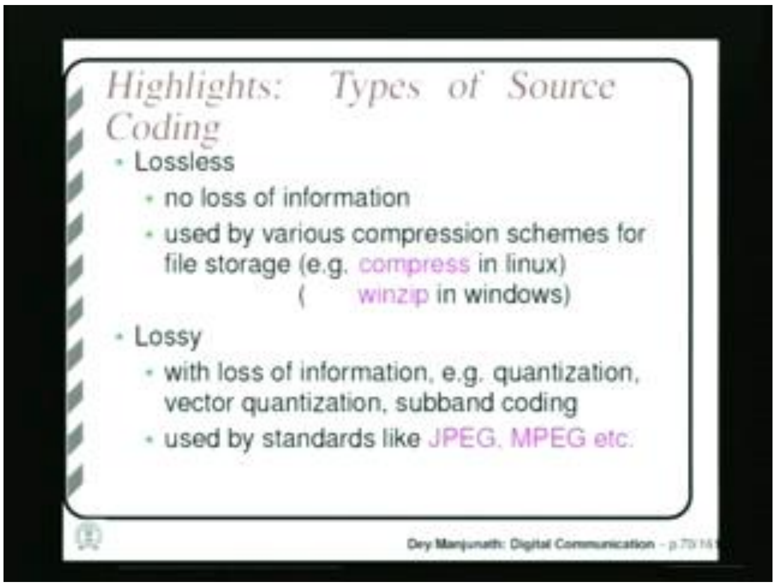
Dey Manjunath: Digital Communication

So, so far in source coding we have covered, we have seen why we should do source coding. And then we have seen, different desirable properties of source codes like: uniquely decidable property, then prefix property and we have seen 1 particular source coding technique: called Huffman code and we have, in the last class we have seen Kraft inequality and we have proved Kraft inequality. Kraft inequality gives us a condition of condition for existence of a prefix code of certain given lengths. So, given certain lengths if, Kraft inequality satisfied by those lengths then, a prefix code exists. Otherwise, a prefix code does not exist with those lengths.

Then, we have investigated into, what kind of parameters, what kind of, how much compression is possible in practice? So, in terms of the entropy of the source, we have seen that the compression, the minimum required number of bits will be in between HX and HX plus 1. And in this class we are going to cover block wise source coding.

So, we will see that using blocks of coding blocks of source symbols together, we can achieve better compression. And then we will discuss another source coding technique called Shannon-Fano-Elias Code.

(Refer Slide Time: 02:39)



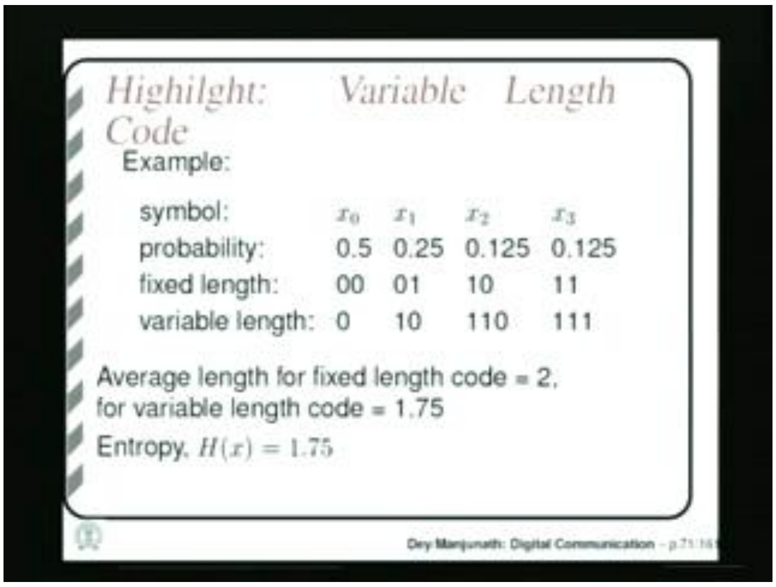
Highlights: Types of Source Coding

- Lossless
  - no loss of information
  - used by various compression schemes for file storage (e.g. compress in linux)
    (winzip in windows)
- Lossy
  - with loss of information, e.g. quantization, vector quantization, subband coding
  - used by standards like JPEG, MPEG etc.

Dey Manjunath: Digital Communication - p.70/16

So, we will first go through what we have done in the previous classes. First we have seen that, there are two types of source coding. One is lossless where there is no loss of information due to coding. So, a typical example of lossless source coding is the; when the file compression algorithms that we use in computer like: compress in Linux and WinZip in windows. And then is another class of source coding techniques called: lossy source coding where, there is some loss of information due to compression. And there are popular standards like: JPEG MPEG for compression in image and video. They are all lossy source coding.

(Refer Slide Time: 03:17)



Highilght: Variable Length Code

Example:

symbol:	x_0	x_1	x_2	x_3
probability:	0.5	0.25	0.125	0.125
fixed length:	00	01	10	11
variable length:	0	10	110	111

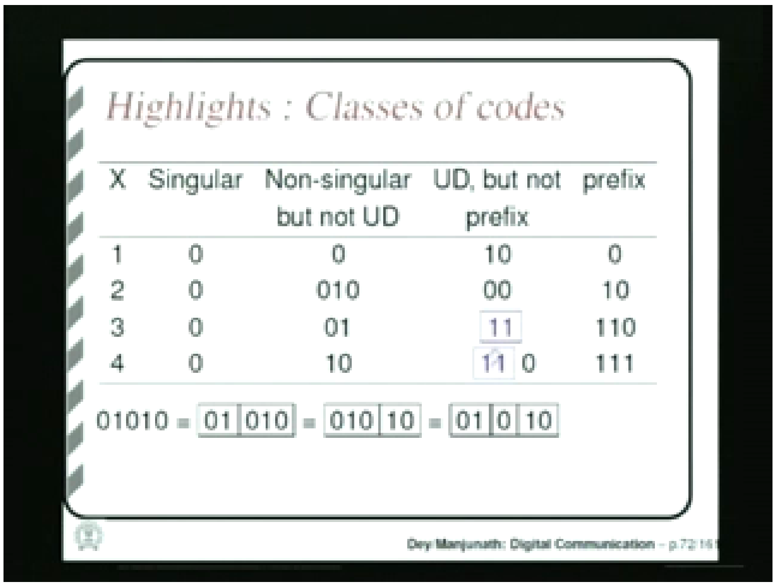
Average length for fixed length code = 2,
for variable length code = 1.75

Entropy, $H(x) = 1.75$

Dey Manjunath: Digital Communication – p.71/161

Then, we have seen that using variable length coding; we can achieve better compression than using fixed length representation of the source symbols.

(Refer Slide Time: 03:19)



Highlights : Classes of codes

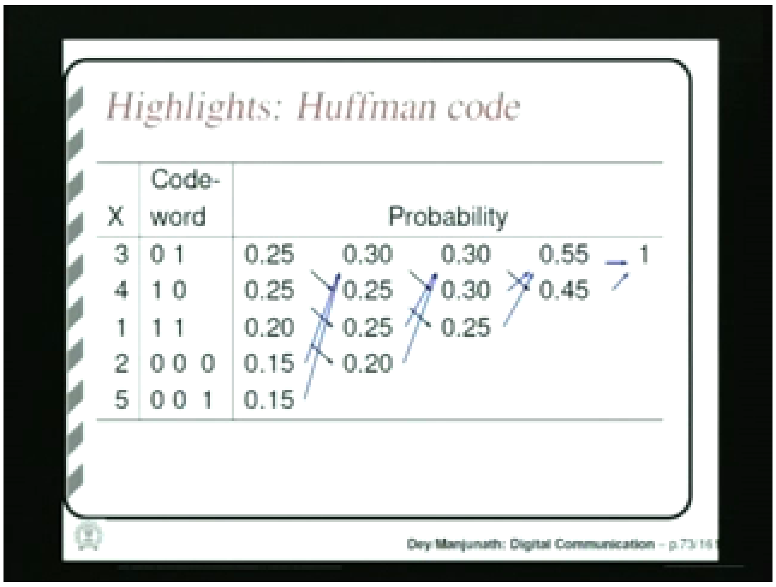
X	Singular	Non-singular but not UD	UD, but not prefix	prefix
1	0	0	10	0
2	0	010	00	10
3	0	01	$\boxed{11}$	110
4	0	10	$\boxed{11}$ 0	111

01010 = $\boxed{01|010}$ = $\boxed{010|10}$ = $\boxed{01|0|10}$

Dey Manjunath: Digital Communication – p.72/161

Then, we have seen that there are different classes of codes: nonsingular, but not uniquely decodable and then uniquely decodable, but not prefix code, but finally, there are prefix codes. So, we actually want prefix codes because, they are the most they are instantaneously decodable. So, this code was not a prefix code, but it is uniquely decodable, whereas, this code is a prefix code.

(Refer Slide Time: 03:59)



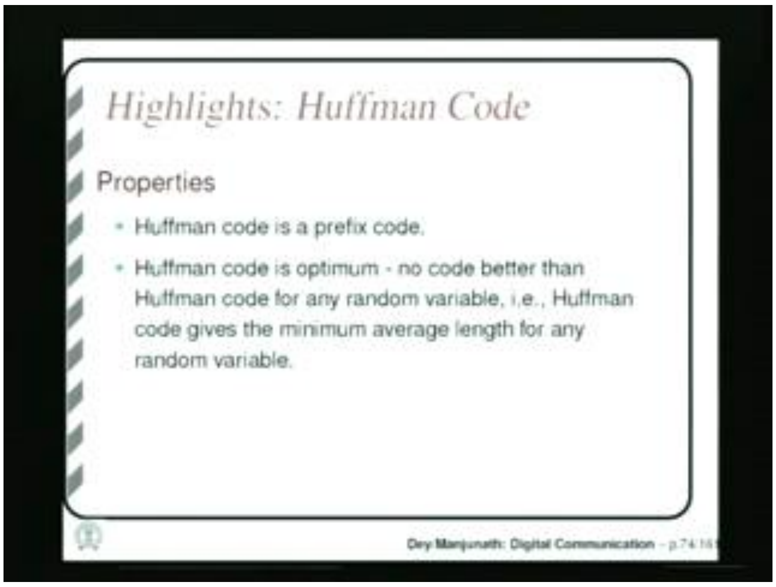
Highlights: Huffman code

X	Code-word	Probability				
3	0 1	0.25	0.30	0.30	0.55	1
4	1 0	0.25	0.25	0.30	0.45	
1	1 1	0.20	0.25	0.25		
2	0 0 0	0.15	0.20			
5	0 0 1	0.15				

Dey Manjunath: Digital Communication – p.73/16

Then we have seen, Huffman coding. We have discussed how to do Huffman coding. And so, first rearrange the probabilities in decreasing order then, combine the last 2 probabilities again and again and assign 0 and 1 to the corresponding symbols.

(Refer Slide Time: 04:19)



Highlights: Huffman Code

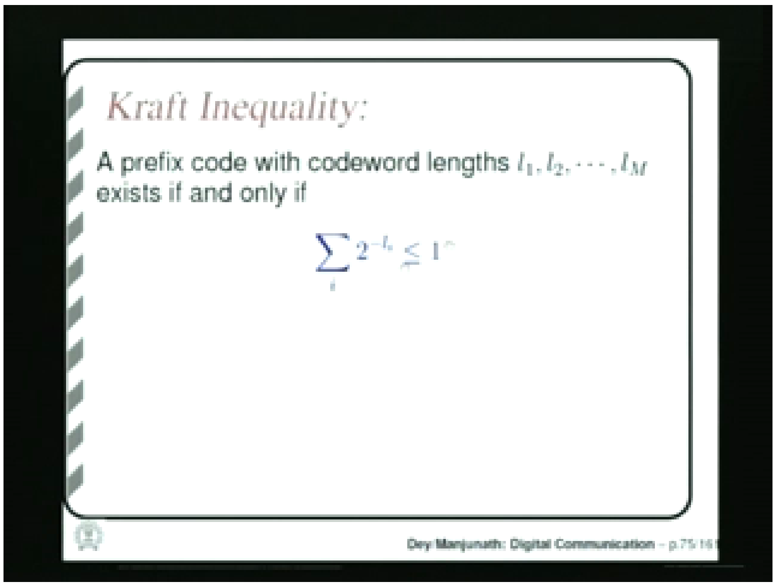
Properties

- Huffman code is a prefix code.
- Huffman code is optimum - no code better than Huffman code for any random variable, i.e., Huffman code gives the minimum average length for any random variable.

Dey Manjunath: Digital Communication – p.74/16

Then Huffman code we have seen that, Huffman code is a prefix code and Huffman code is also optimum; in the sense that there is no code better than Huffman code in terms of average code length part symbol. So, Huffman code is a, is an optimum code. So, its lengths will be between Hx and Hx plus 1; average length.

(Refer Slide Time: 04:44)



Kraft Inequality:

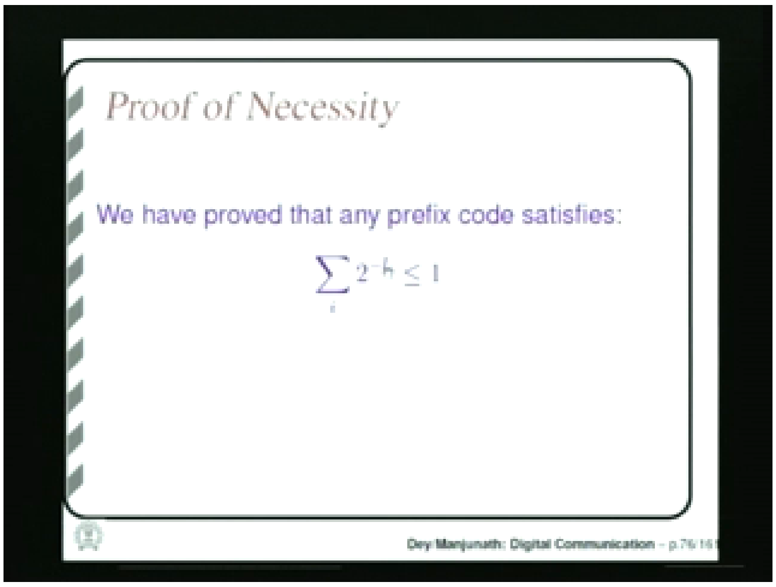
A prefix code with codeword lengths $l_1, l_2, \cdots, l_M$ exists if and only if

$$\sum_i 2^{-l_i} \leq 1$$

Dey Manjunath: Digital Communication – p.75/161

Then, we have seen we have discussed Kraft inequality. The Kraft inequality says that: a prefix code with this given lengths exists, if and only if this condition is satisfied by these lengths. And we have proved both the parts necessary necessity and sufficiency conditions in the last class. So, necessity says that, this condition is necessity for existence of a prefix code, naming by prefix code with these lengths, should satisfy this condition.

(Refer Slide Time: 05:14)



Proof of Necessity

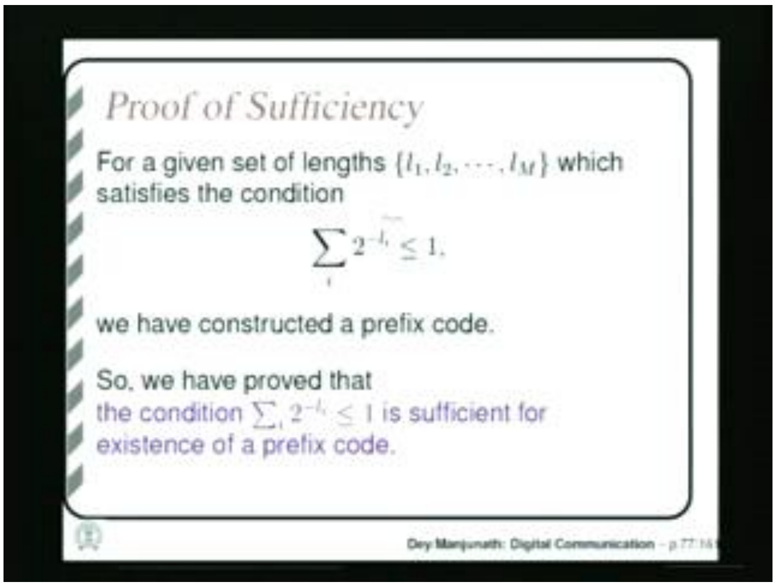
We have proved that any prefix code satisfies:

$$\sum_i 2^{-l_i} \leq 1$$

Dey Manjunath: Digital Communication – p.76/161

So, we have proved necessity. We have proved that any prefix code satisfies this condition.

(Refer Slide Time: 05:21)



Proof of Sufficiency

For a given set of lengths $\{l_1, l_2, \cdots, l_M\}$ which satisfies the condition

$$\sum_i 2^{-l_i} \leq 1,$$

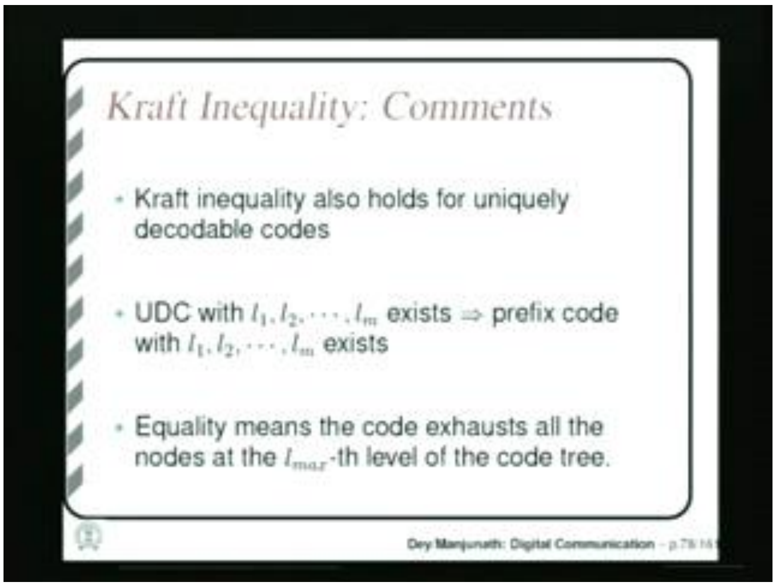
we have constructed a prefix code.

So, we have proved that the condition $\sum_i 2^{-l_i} \leq 1$ is sufficient for existence of a prefix code.

Dey Manjunath: Digital Communication - p.77/161

And then, we have proved sufficiency any way, this condition is also sufficient for existence of a prefix codes of these length. And that we have proved by constructing a prefix code with these lengths. If a set of lengths is given which satisfy this condition then, we have proved that there is a prefix code with these lengths. So, this condition is sufficient for existence of a prefix code.

(Refer Slide Time: 05:38)



Kraft Inequality: Comments

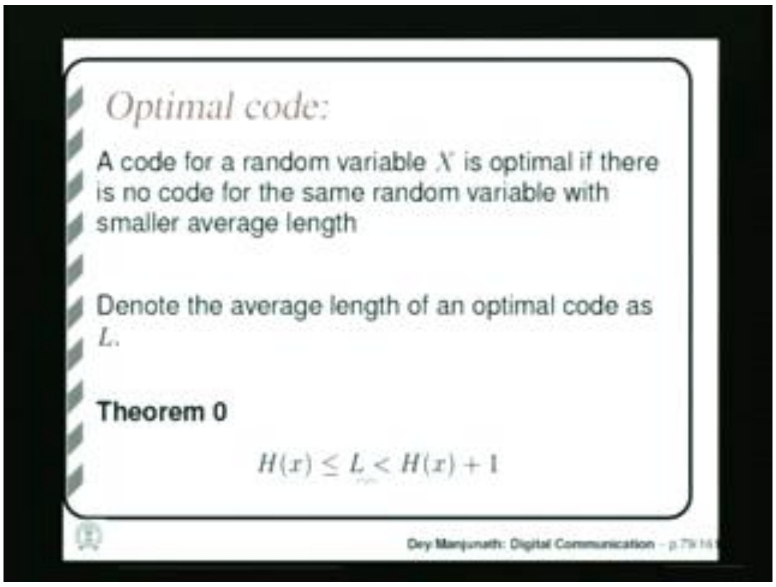
- Kraft inequality also holds for uniquely decodable codes
- UDC with $l_1, l_2, \cdots, l_m$ exists $\Rightarrow$ prefix code with $l_1, l_2, \cdots, l_m$ exists
- Equality means the code exhausts all the nodes at the l_{max}-th level of the code tree.

Dey Manjunath: Digital Communication - p.78/161

So, Kraft inequality we also made comments that, Kraft inequality also holds for uniquely decodable codes. And that means: that whenever there is uniquely decodable code with these lengths, there is also prefix code with these lengths. Because this: a uniquely decodable code with these lengths exist means; that these lengths satisfy Kraft inequality because, Kraft inequality also holds for uniquely decodable codes. And because these lengths satisfy uniquely Kraft inequality, a prefix code also exists with these lengths.

So, this comment means: that we should always use prefix code because, whenever there is uniquely decodable code with certain lengths, we can also find a prefix code with those lengths. So, we will rather use that prefix code because, we can then decode in the code instantaneously. Also we have seen that, equality in the Kraft inequality means: that the code exhausts all the nodes, at the highest level of the code tree. That we have seen in the last class.

(Refer Slide Time: 06:44)



Optimal code:

A code for a random variable X is optimal if there is no code for the same random variable with smaller average length

Denote the average length of an optimal code as L.

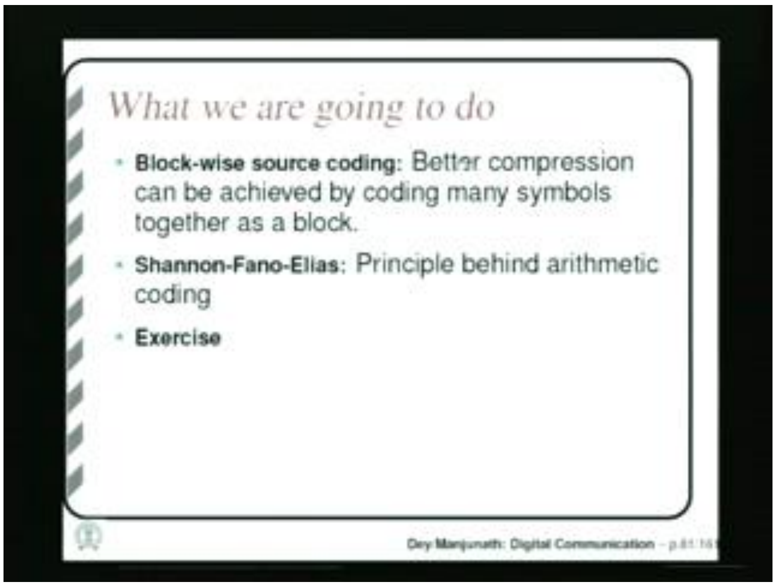
Theorem 0

$$H(x) \leq L < H(x) + 1$$

Dey Manjunath: Digital Communication – p.79/

And then we have discussed what is optimal code. A code will be called if there is, no better code for the same random variable; better in the sense that, there is no code with smaller average length. So, if an optimal code has length L then, we have proved in the last class that, L will lie between Hx and Hx plus 1. So, Huffman code; Huffman codes average length will certainly lie between this range means: because Huffman code is an optimal code. So, this is what we have done in the last class.

(Refer Slide Time: 07:17)



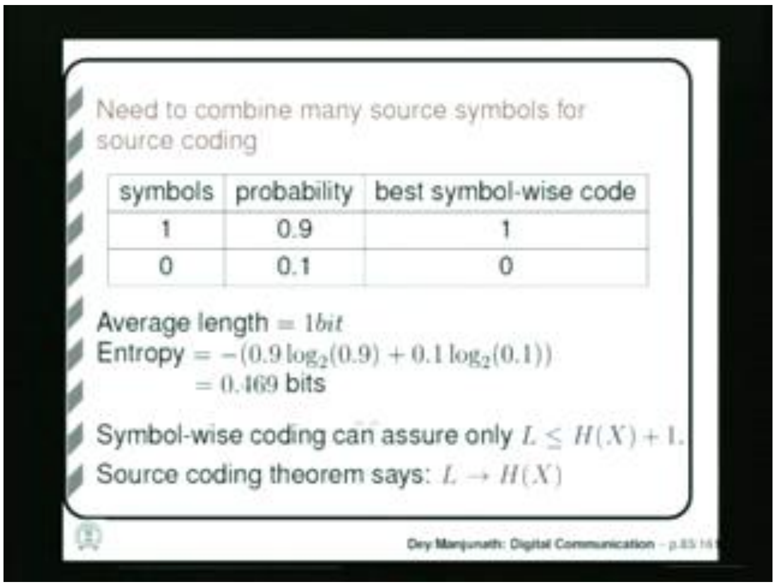
What we are going to do

- **Block-wise source coding:** Better compression can be achieved by coding many symbols together as a block.
- **Shannon-Fano-Elias:** Principle behind arithmetic coding
- **Exercise**

Dey Manjunath: Digital Communication – p.81/16

Now, in this class we are going to do block wise source coding. So, we will see that better compression can be achieved, by coding many symbols together as a block. And then, we will do we will discuss 1 another source coding technique called Shannon-Fano-Elias code. And this is also a very important code because this actually gives the principle for arithmetic coding; a very again very important source coding technique. And then we will solve some exercise in this class.

(Refer Slide Time: 07:46)



Need to combine many source symbols for source coding

symbols	probability	best symbol-wise code
1	0.9	1
0	0.1	0

Average length $= 1 bit$
Entropy $= -(0.9\log_2(0.9) + 0.1\log_2(0.1))$
$= 0.469$ bits

Symbol-wise coding can assure only $L \leq H(X) + 1$.
Source coding theorem says: $L \to H(X)$

Dey Manjunath: Digital Communication – p.83/16

So, let us start with block wise source coding. Let us first see that coding symbol wise is now, not always the best thing to do. Suppose, we have a binary symbol takes values 1 or

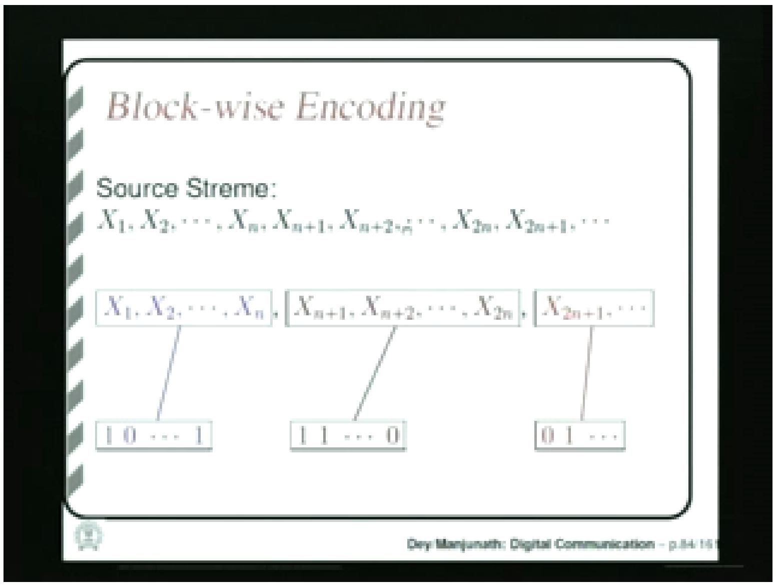
0. It takes value 1 with probability 0.9 and 0 with probability 0.1 suppose. Now, what is the best symbol wise code for this random variable? The code can is 1 0, you have no other choice. You need at least 1 bit part symbol to represent the random variable and so, only choice you have is 1 0. But this will require 1 bit of average length whereas, the entropy of the source is 0.469 bits.

So, this is also the Huffman code; this is the optimal code for this random variable. So, this is not really giving us any compression. So, symbol wise coding can assure, we have seen that this average length: length less than equal to Hx plus 1; where in this case Hx is this, we are achieving average length 1 which is between this and this plus 1 as we see, between 0.469 and bit and 1.469.

So, source coding theorem on the other hand says: that you can always code a random variable such that, the average length goes tends to Hx meaning by; if Hx is this then, we can code this random variable with average length as close to this as we want. Meaning by: suppose we want entropy is this so it cannot, average length cannot be less than this, but suppose we want 0.4 0.47, we want average length 0.47. Source coding theorem says that, there is a source code for this random variable, which will achieve average length 0.47, because 0.47 is greater than this.

So, any length greater that Hx can be achieved by a source code. But, how to do that coding? We see that with symbol wise coding that is, if you code 1 symbol at a time independently then, there is a, this is not possible average length the best average length you can get is 1 bit.

(Refer Slide Time: 10:12)



Block-wise Encoding

Source Streme:

$X_1, X_2, \cdots, X_n, X_{n+1}, X_{n+2}, \cdots, X_{2n}, X_{2n+1}, \cdots$

$\boxed{X_1, X_2, \cdots, X_n}, \boxed{X_{n+1}, X_{n+2}, \cdots, X_{2n}}, \boxed{X_{2n+1}, \cdots}$

$\boxed{1\ 0\ \cdots\ 1}$ $\boxed{1\ 1\ \cdots\ 0}$ $\boxed{0\ 1\ \cdots}$

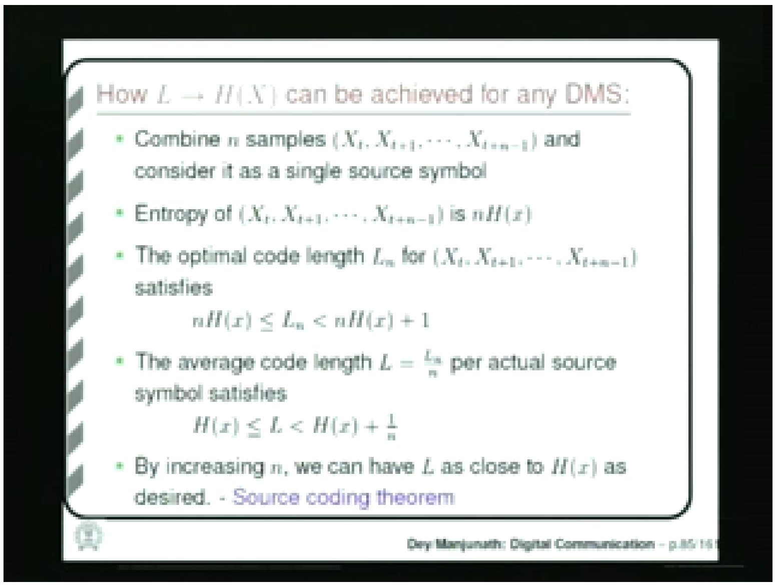
Dey Manjunath: Digital Communication – p.84/161

So, that can be done, this can be achieved by doing block wise source coding. What is the idea? The idea is that, the source is generating all these symbols 1 after another. These are all identical and independent random variables and independent and identically distributed random variables. In short it is called IID. So, these are all IID. Now, you instead of coding X 1 and X 2 and X 3 independently, we code X 1 to X n, we are taking a block of n symbols together. Then, next block of n together, next block of n together this way and then, coding this block of n symbols together into 1 bit stream.

So, how will I, how will you do the coding? We will take this whole block of random variables as a single random vector; a random variable. So, it will take many different values for example, if X 1 X 2 each is binary then, this will take 2 to the power n possible values, all possible n bit patterns. So, then they will all have some probability, each n bit pattern will have some probability based on the provability X 1, distribution of X 1 X 2 etcetera.

So, we can find out those probabilities and then, do coding; source coding for this block of n symbols together. So, this block will give us 1 code word, this block will give us another code word and so on and this will be our code stream. So, if we do coding this way, we can see that we can get better compression and we will this in a while.

(Refer Slide Time: 12:05)



How $L \to H(X)$ can be achieved for any DMS:

- Combine n samples $(X_t, X_{t+1}, \cdots, X_{t+n-1})$ and consider it as a single source symbol
- Entropy of $(X_t, X_{t+1}, \cdots, X_{t+n-1})$ is $nH(x)$
- The optimal code length L_n for $(X_t, X_{t+1}, \cdots, X_{t+n-1})$ satisfies
  $$nH(x) \leq L_n < nH(x) + 1$$
- The average code length $L = \frac{L_n}{n}$ per actual source symbol satisfies
  $$H(x) \leq L < H(x) + \frac{1}{n}$$
- By increasing n, we can have L as close to $H(x)$ as desired. - Source coding theorem

Dey Manjunath: Digital Communication - p.85/161

So, this is how, using block wise source coding we can achieve for, we can achieve this average length; average length tending to HX for any discrete memory less source. So, what is the idea? We have a source sequence; source is generating several random variables, a sequence of random variables 1 after another. Now, instead of coding 1 symbol at a time, we are taking a block of n symbols together, next block of n symbols together and so on. And finding out the probability distribution, of a block of n symbols, not every symbol independently. So, for binary random variable, if you take say 4 bits at a time then, every 4 bit pattern 1 1 0 1 will have some probability, because 1 will have probability and 0 will have some probability. So, you can compute the probability of 1 1 0 1 0 0 1 0 etcetera. So, we can find out the probability distribution of the n bit patterns and then, do source coding of n symbols together. So, by doing that, we can actually achieve better compression, as we see in this slide.

Suppose, we combine n symbols together like this: X t to X t plus n minus 1, there are n symbols here and consider it as a single source symbol. Then, entropy of this vector random vector is n times HX, because each has entropy that is average information HX. So, and all of them are independent and identically distributed. So, the entropy of n symbols together will be n times HX. Then, if we find an optimal code for this random vector; so this is a, this is like a single symbol now. We can find a random vector, we can find an optimal code for this random vector and we can, we will get suppose a length L n. Then, we know from what we did in the last class that, this L n will be in between entropy of this and entropy of this plus 1. And what is the entropy of this random vector? It is n times HX.

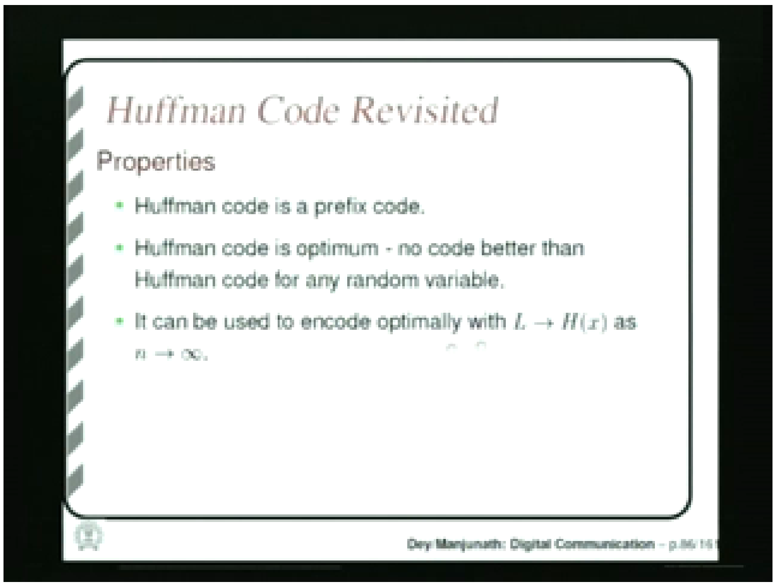
So, this optimal code length will be in between n times HX and n times HX plus 1. Now, part symbol what is the code length we have used? Average code length part symbol is now, L n by n because for n symbols we have L n number of bits, to represent this n symbols. So, part symbol on average we have, L n by n number of bits. So, now if you divide all these terms by n, what do you get? We get HX is less than equal to L because, L n by n is L. That is the average code length part symbol, for actual source symbol this. So, we get HX is less than equal to L less that equal to HX plus 1 by n.

So, now part symbol what is the code length we have got? We have got between HX and HX plus 1 n. Previously by doing symbol wise coding we got 1 here. So, now, this length will be certainly nearer to HX than what we could get by doing symbol wise coding, because that was between HX and HX plus 1 and this is between HX and HX plus 1 by n, 1 by n is smaller than 1.

So, it is closer to HX. So, by increasing n we can see that, L can be taken as close to HX as we wish. Because by increasing n this will, this side also will tend to HX. So, L will tend to HX because, it is in between HX and HX when n tends to infinity. So, L will tend to HX. So, this in fact, this actually will give us the limiting optimum compression that is, given by source coding theory. So, L will tend HX as n tents to infinity.

So, what we have done is; by taking blocks of symbols together, we have seen that the optimal code length part symbol will be in between HX and HX plus 1 by n. So, as n tends to infinity, so as we increase the block length, we will get the average code length part symbol will tend to HX. So, by increasing the block length, we actually go, we actually achieve a average code length nearer and nearer to the entropy of the source. So, that is what source coding theorem says; that it is possible to do so and we have seen how to do it.

(Refer Slide Time: 17:16)



Huffman Code Revisited

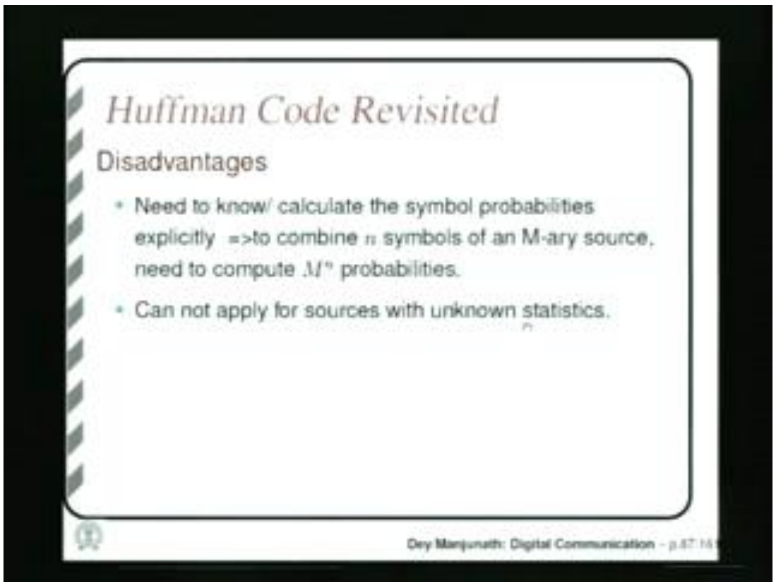
Properties

- Huffman code is a prefix code.
- Huffman code is optimum - no code better than Huffman code for any random variable.
- It can be used to encode optimally with $L \to H(x)$ as $n \to \infty$.

Dey Manjunath: Digital Communication – p.86/161

Then, we have these are, these we have already discussed. Now, from previous slide we also see that; Huffman code can be used to encode optimally with L tends to HX as n tends to infinity; that means, we can instead of doing Huffman coding for every symbol now, we can combine n symbols together and do Huffman coding for blocks of n symbols. And then, as we increase n our Huffman code will give us L tends to HX; part symbol part symbol the average code length will tend to HX. So, that can be achieved by Huffman code itself because, Huffman code is optimal.

(Refer Slide Time: 18:04)



Huffman Code Revisited

Disadvantages

- Need to know/ calculate the symbol probabilities explicitly =>to combine n symbols of an M-ary source, need to compute M^n probabilities.
- Can not apply for sources with unknown statistics.

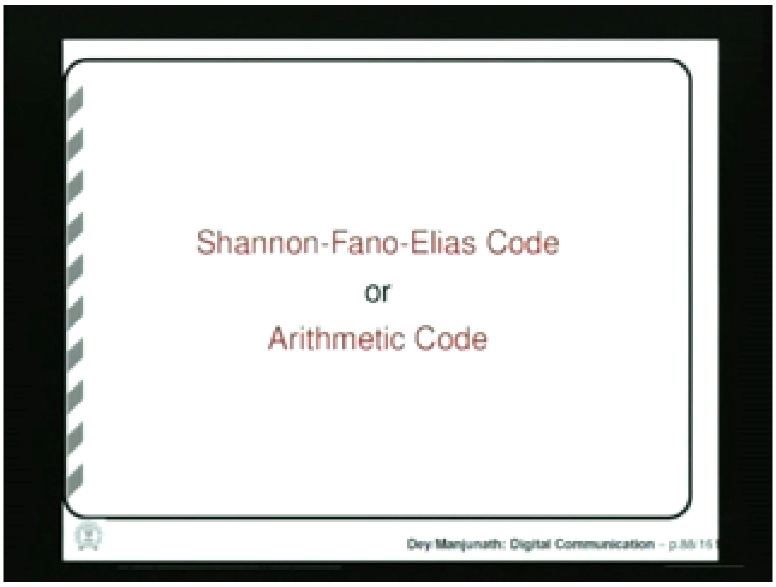
Dey Manjunath: Digital Communication – p.87/161

Now, what is the disadvantage of Huffman code? 1 disadvantage is that Huffman code requires computation of the probability mass function, for every possible symbol values.

So, if we have a block of n symbols, we are doing Huffman coding for a block of n symbols together and every symbols suppose takes m values, m may be 2 for binary source. Every symbol takes m values then we have n number of symbols together. So, what is the possible, what is the number of possible values for n length block? It is m power n. So, m values for each symbol that power the block length. That is the number of values the block of n symbols can take.

So, we need to compute m power n number of probabilities to do Huffman coding and that is a huge task if m is large, we are saying, we are talking about n tends to infinity. So, we are going to take large block length. So, that is a huge task to compute all the probability mass all the probabilities. Then, we cannot apply this techniques Huffman code for sources with unknown statistics. There are practical sources for which the statistics is not known beforehand. So, we cannot apply Huffman coding for that kind of sources. So, now we will discuss 1 another technique; source coding technique called Shannon-Fano-Elias Code or it is also called Arithmetic Code.

(Refer Slide Time: 19:34)



Shannon-Fano-Elias Code

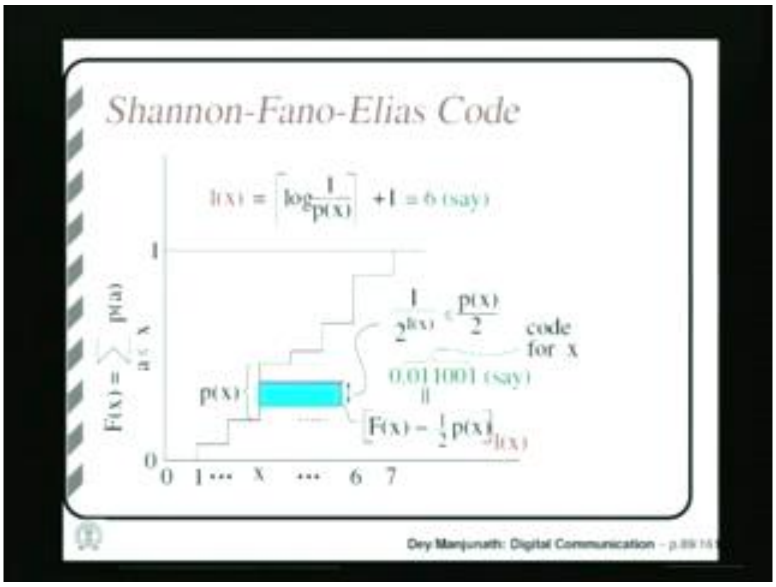
or

Arithmetic Code

Dey/Manjunath: Digital Communication – p.88/16

It is in fact, this is a, this is this gives us the principle behind arithmetic coding. We will see arithmetic coding in the next class, but we will discuss the principle behind it in this class. So, Shannon-Fano-Elias Code.

(Refer Slide Time: 19:52)


Shannon-Fano-Elias Code

$l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil + 1 = 6$ (say)
$F(x) = \sum_{a \le x} p(a)$
1
0
$p(x)$
$\frac{1}{2^{l(x)}} < \frac{p(x)}{2}$
code for x
0.011001 (say)
$\lfloor F(x) - \frac{1}{2} p(x) \rfloor_{l(x)}$
0 1 ⋯ x ⋯ 6 7
Dey Manjunath: Digital Communication

How do we go about it? Suppose the source symbol takes; source random variable takes so many values. In this case we have taken 8; 8 values 0 1 to 7. We write them in some sequence, in some order and then we know; what is the cumulative distribution function of the random variable. It is defined this way: cumulative distribution function the value at 3 for example, is nothing, but the probability that the random variable takes value less than equal to 3.

So, probability of 0 plus probability of 1 plus probability of 2 plus probability of 3 that is the value here. So, if you plot a cumulative distribution function of a discrete random variable, you will get a staircase function like this. It is, it will be a, an increasing function, because at every place it can only increase; it is the sum of the probabilities of previous values. So, it will increase and the highest value will be 1; it will saturate to 1, it will start from 0 and it will go to 1. This is the; this is the cumulative distribution function of the random variable.

Now, suppose we have some value X and we; obviously, this, this jump is the probability of X because, at X the value is here, at X minus 1 the value is here. So, the probability this is nothing, but the cumulative distribution function of X, is sum of the probability till X and whereas, the value here is sum of the probability still X minus 1. So, what has increased here is only the probability of X. So, this jump is nothing, but probability of X. So, this is probability of 1 this, this amount, this quantity is probability of 1, this quantity is probability of 2 and so on.

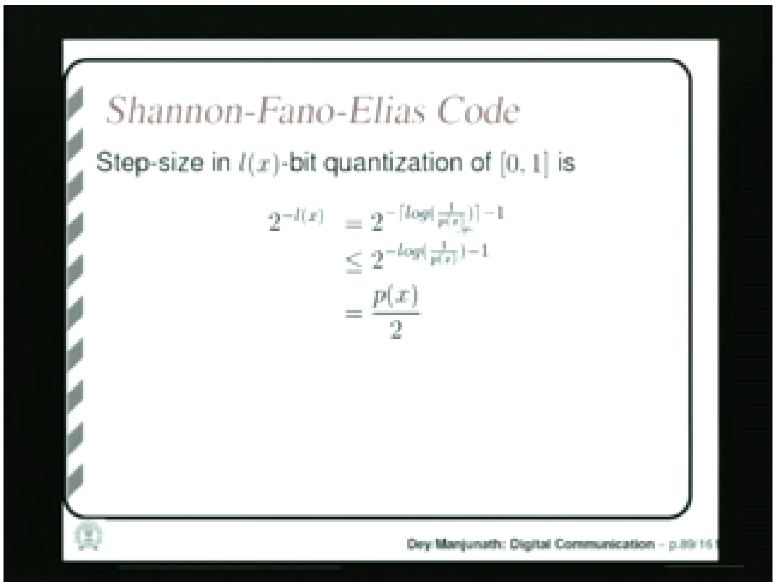
Now, we will see how to encode to value X. If the random variable takes X the value X what will be the code word? We will give a way of computing the code word. So, we have seen that this is F x; the cumulative distribution function value at x, this is the value at x minus 1 that is nothing, but F x minus p x. p x is the probability of x. And we see, suppose, we take the midpoint here that is F x minus half of p x, this is the midpoint of this jump; then, we compute this quantity. We know p x, we compute log of 1 by p x. And we have seen before hand, that to do to design a good code, we should in fact, take lengths which are proportion which are same as which is, same log of 1 by p x.

Now, we are not taking exactly that, because this may be fraction. So, what we do is, we take the first, we take the ceiling of this ceiling means: again the smallest integer above this number. So, like if this number log of 1 by p x is 2.5, the ceiling will be 3, if it is 3.9 that ceiling will be 4. So, this ceiling plus 1 we increase it by 1. So, that is the length we take for the code for x; for the code word for x we take this length.

So, it is slightly more than what we should take for optimal codes. And then we express this quantity as a binary sequence. So, this is the F x minus half of p x, this number this will be a fraction 0 point something, because it is between 0 and 1. So, it will be 0 point something, but we express that in as a binary number. So, when we express it as a binary number, what will we get? We will get some binary sequence of a point; so 0 110110011 like that. So, some binary sequence after the point we will get. So, that is the binary representation of that number.

Now, we take only L x number of bits after the point. So, that number is denoted here by, this; this number quantizes to X bits. So, we take only L x number of bits after a point. So, we basically truncate this number, after L x number of bits. So, we have suppose, L x equal to 4 then, that number is 0.11011100. Then what we do? We truncate this number to 4 bits. So 0.1101, that is the new number; certainly that is less than the original number. So, this number is less than this because, this is the truncated number. So, this is somewhere here and we will see that this cannot go below this.

(Refer Slide Time: 25:21)



Shannon-Fano-Elias Code

Step-size in $l(x)$-bit quantization of $[0,1]$ is

$$\begin{aligned} 2^{-l(x)} &= 2^{-\lceil \log(\frac{1}{p(x)})\rceil - 1} \\ &\le 2^{-\log(\frac{1}{p(x)})-1} \\ &= \frac{p(x)}{2} \end{aligned}$$

Dey Manjunath: Digital Communication – p.89/161

In the next slide, we will see that this number will not go below this and because, when we take a less number of bits, we are basically dividing the, this number this interval 0 to 1 into 2 power l x number of parts. Because, how many l x bit combinations are there? 2 power l x; so many combinations are there. So, we are taking 1 number from this interval, 1 number from those 2 power l x divisions of this interval and basically this part is nothing, but the 1 level that is nearest to this below this.

So, we divide this interval into 2 power l x number of parts and immediately below this, there will some level, we take that level that is, this number that will be this number.
So, what is the, this difference cannot be more than the each size of each quantize level. So, we have 2 power l x parts here. So, 1 by 2 power l x is the height of each section. So, this difference cannot be more than that minus that 1 by 2 power l x that is 2 power minus l x.

So, and 2 power minus l x is equal to we can see, that l x is this ceiling of this number plus one. So, you replace that l x by this and then we see that, ceiling is certainly greater than this number. So, this number will be less that if you remove the ceiling, because this quantity is greater less than this quantity. So, minus of that is greater than this quantity. So, this whole number is greater than this. And this is 2 power minus log 1 by p x minus 1, minus 1 is nothing, but half and what is 2 power minus log 1 by p x? Minus log 1 by p x is log p x and 2 power that is just p x. So, it is p x by 2.

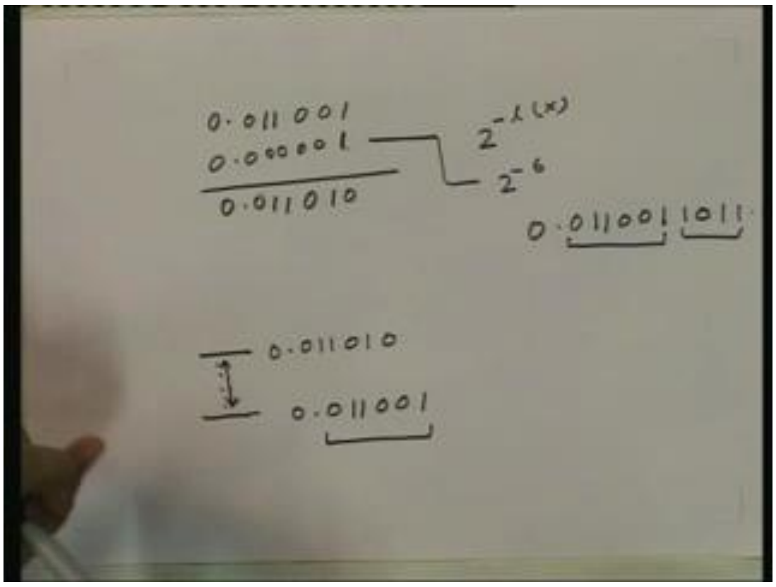
So, this number is less than equal to p x by 2 whereas, this difference is p x by 2 this difference. So, this difference the quantity by which this number is reduced by taking only a only l x bits that difference will be, less than equal to the step size of the quantization that is, 2 power minus l x and 2 power minus l x is less than equal to p x by 2. So, this difference is less than equal to p x by 2. So, this will not go below this level that is idea. So, we know that this number is in between this range.

So, we have seen how to take the code. Basically represent this number in a binary form and then take l x number of bits after the points point and so in this case say: suppose l x is 6 and suppose this is this, with some 6 bits. We have taken 6 bits of the original number here. And so, this is the code for x; this 6 bits after point 011001. We are taking 6 bits of the; of the midpoint here.

So, we have seen how to get the, how to construct the code word for x. So, for every value here, we will take the code words. So, for 1 we will get 1 code word, for 0 we will get, for every place, for every value we will get a code word because, we know the cumulative distribution function and these probabilities. Now, we have seen how to get the code words. Now we need to see, we need to get convinced that, this is uniquely decodable and in fact, we will show that this code is a prefix code.

So, let us proceed towards that. We have this number we will show that, this code we have obtained this way will be a prefix code. We have got this; now, we add 1 to the last bit of this.

(Refer Slide Time: 30:40)



So, we have point something, something some number. So, we have this number and then, suppose this number is point we have taken 0.011001 this number. We add 1 to the last bit. So, what do you get? We get 010110. So, we construct another number. So, what have we done? We have added basically like this, we add 1 to last bit meaning by: we add this quantity to this quantity.

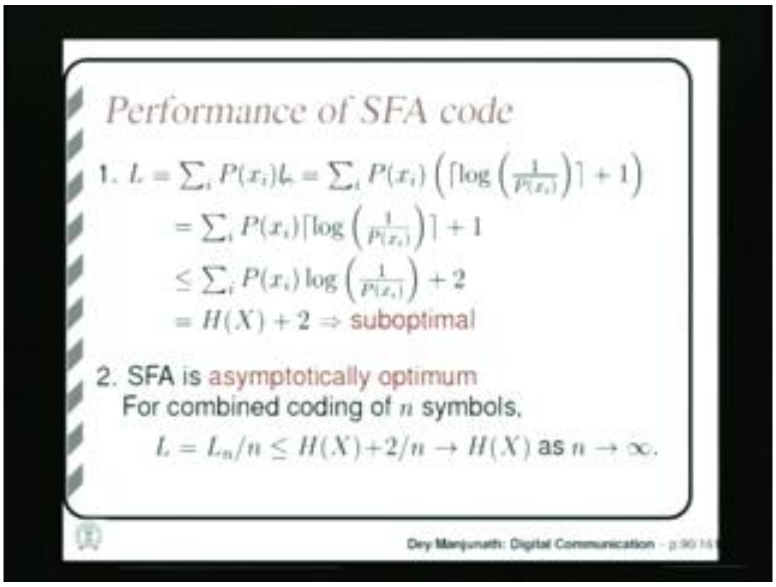
So, what is this quantity? What is the value of this? This is nothing, but 2 power minus l x that is in this case, it is 2 power minus 6 because, value of first bit is 2 power minus 1, value of the first second bit after point is 2 minus 2 etcetera, and so, the value of the l x number of bit, at position l x after point is 2 power minus l x. So, we have added 2 power minus l x to the, to this quantity. So, and we added 2 power minus l x and we have already seen that, 2 power minus l x is less than equal to p x by 2. So, the new number will be also below this level, because this difference is greater than equal to p x by 2. So, we have added 2 power minus l x to this quantity which will be below this number. So, this is the new number we get.

Now, observe 1 thing that, we have we first had 1 level that is this 1; 0.011001, then we added 1 to the last bit, we get another level that is this 0.011010. We have added 1 to the last bit and we have got this. Now, all the numbers of which, this part is the prefix; suppose, we take 0.011001 and then some other bits 1011 extra some extra bits then, this number because it starts with this number, this number will lie somewhere here because, this number will be between this number and this number; because this is certainly less than this number because, this number is obtained by adding 1 to this bit.

So, any number of which, this is a prefix will be somewhere here in this interval. So, meaning by: any number of which this number is a prefix will be in this range, in this interval. So, we get such an interval for every possible symbol values. We will get 1 interval here, 1 interval here. Now, we can see clearly here that, this code will be prefix code because, the code word we obtained here this is not a prefix of this or this is not a prefix of this because, this number does not belong in this region. So, this is, this cannot be a prefix of that because, all the number of which this prefix is in this range. Similarly, this is also a not yet prefix of this number.

So, the code words we get this way will be a prefix code. So; obviously, it will be an uniquely decodable code. So, just now what I was saying; we add 1 to this bit, we get this number. So, all the numbers in between this range are the numbers of which, this is prefix or other numbers cannot be a prefix of; this cannot be a prefix of any other number which is not in this range. So, this way we can argue that this code will be a prefix code.

(Refer Slide Time: 34:41)



Performance of SFA code

1. $L = \sum_i P(x_i) l_i = \sum_i P(x_i) \left(\left\lceil \log \left(\frac{1}{P(x_i)} \right) \right\rceil + 1 \right)$
$= \sum_i P(x_i) \left\lceil \log \left(\frac{1}{P(x_i)} \right) \right\rceil + 1$
$\leq \sum_i P(x_i) \log \left(\frac{1}{P(x_i)} \right) + 2$
$= H(X) + 2 \Rightarrow$ suboptimal

2. SFA is asymptotically optimum
For combined coding of n symbols,
$L = L_n/n \leq H(X) + 2/n \rightarrow H(X)$ as $n \rightarrow \infty$.

Dey Manjunath: Digital Communication

Now, something interesting; we have so far seen that, the code is prefix code. So, what else are you interested in, in a code? We have to check that it is uniquely decodable. So, we have seen that because it is prefix code. Then we have see, what is the compression, how much compression we get using that code, meaning by: what will be the average length of the code words? And what is that? We have to basically to take the average of all the possible code words. So, that is what we do here. We take the average length; this is summation of p x i l i; l i is the length of ith code word. Now, l i we have seen is nothing, but this. We have taken l i that way.

Now, we take summation with bracket here. So, p x i times this plus 1 because, summation p x i is 1. Now, this ceiling is less than equal to this quantity plus 1 because, if this number is 2.5; ceiling is 3. So, 3 is less than 2.5 plus 1 that is, 3.5 certainly between any number. And if you add 1 to it, between those 2 numbers there will be 1 integer. So, that integer next to any number will be less that number plus 1. So, this is less than log 1 by p x i plus 1. So, we have done that.

So, another 1 comes out of this summation because, summation p x i greater than 1. So, we get 2 here, 2 plus this. And what is this quantity? This is nothing, but HX, this is the formula for entropy of the source. So, this is HX plus 2. So, certainly this is suboptimal because, we do not we do not know whether this will be less than HX plus 1 also. What we have granted is that, l will be less than equal to HX plus 2.

So, this code is possibly suboptimal. This is suboptimal, but this not so bad because, now if we suppose combine many symbols together as we have discussed before then, let us see what code length we get, part symbol on average. So, now suppose we combine n symbols and do Shannon-Fano-Elias Code for a block of n symbols together. Then, the average code length l will be l n by n because this is for n symbols. So, divide by n is the part symbol code length. This is less than equal to HX plus 2 by n because, l n is less than equal to n times HX plus 2. Because n times HX is a entropy of n symbols together.
So, the average code length l n for n a block of n symbols will be less than equal to HX n times HX plus 2 because, entropy is n times HX. So, l which is l n by n will be less than equal to HX plus 2 by n. We divide this l n by n n HX by n is HX and then 2 by n. And this will tend to HX as n tends to infinity, this is the interesting part. So, if we instead of coding symbol wise if, we take n symbols together the average code length we get part symbol will be, HX plus 2 by less than equal to HX plus 2 by n.

So, now if we take n tends to infinity meaning by; if as we increase the block length the Shannon Fano Elias Code will code will give us, a average code length part symbol tending to HX. And that is that is good because it says that, using Shannon-Fano-Elias Code also we can achieve what is granted by source coding theorem. So, it is asymptotically optimal as n tends to infinity, we actually get optimal coding l tends to HX.

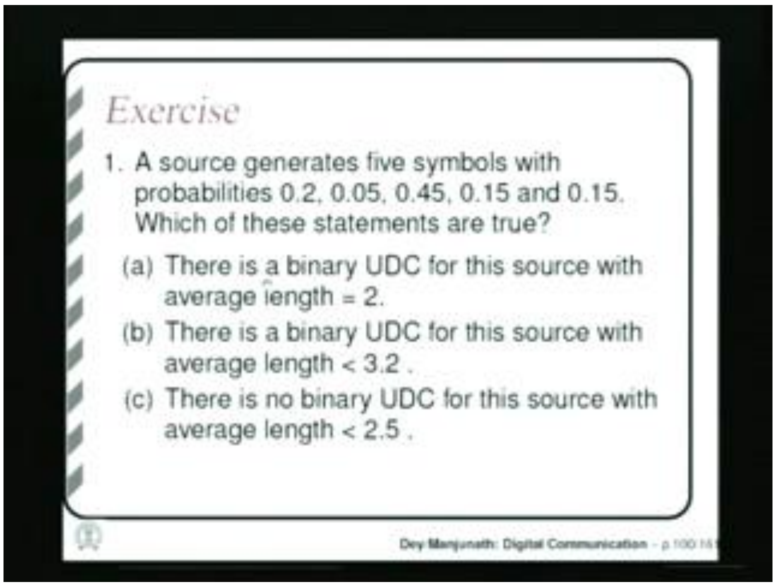
So, we have done in this class: block wise source coding, why we need block wise source coding. With an example we have seen that, there are certain sources for which using just symbol wise coding, we get no compression. Whereas, we can do block wise coding and block white block wise coding will give us; for any source we can get this optimal code length tends to HX. This we can achieve for any discrete memory less course using block wise coding. And we have seen that, because Huffman code is optimal code we can achieve this limit using Huffman code itself, but Huffman code has certain disadvantages, 1 is that it can it needs to compute n power n probabilities that is something huge task and we cannot apply this technique for sources with unknown statistics.

Now, so far we have also discussed how to do Shannon-Fano-Elias Code. We have taken first the cumulative distribution function; this jump here is the value of probability of x at any x. If we take 6; this jump is a probability of the value six. So, we have seen how to construct the code word for x similarly for every value we can construct code word. We take first the midpoint then, take binary representation of the midpoint here and take only l x number of bits, l x is computed this way. Then this will be a number below this because, that is truncated this number is truncated after l x bits and then this take l x number of bits, those l x number of bits here. This is the code for x. And we have seen that this coding will give us a prefix code ah.

Now, this is, this also so far as we see, will require computation of p x and f x for every possible value. So, what disadvantages we have talked about for a Huffman code is also valid so far is this coding. But we will see in the next class, how this can be avoided and how this technique in general can be applied to sources with unknown statics also, some modeling of the statics can be done. And we have seen that, Shannon-Fano- Elias Code is suboptimal in general, but is asymptotically optimal meaning by: if you combine n symbols together and as n tends to infinity we will achieve a code length part symbol tends to HX with Shannon-Fano-Elias Code also; just like Huffman codes.

Though, Huffman code will be better if we do symbol wise coding. For any block of n symbols probably Huffman code, Huffman code is optimal, but Shannon-Fano-Elias Code is not optimal, but as n tends to infinity they will both perform similarly. They will both achieve, they will both tend to HX. So, that is something good for Shannon-Fano-Elias Code also.

(Refer Slide Time: 42:10)



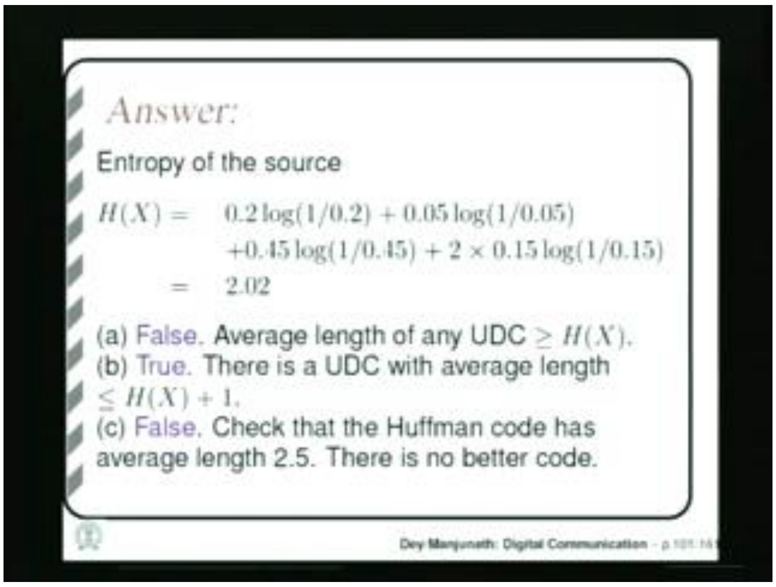
Exercise

1. A source generates five symbols with probabilities 0.2, 0.05, 0.45, 0.15 and 0.15. Which of these statements are true?
   (a) There is a binary UDC for this source with average length = 2.
   (b) There is a binary UDC for this source with average length < 3.2 .
   (c) There is no binary UDC for this source with average length < 2.5 .

Dey Manjunath: Digital Communication - p.100/161

This is a, this is what we have done in this class. Now we will solve some exercise. So, this exercise is that, it says a source generates 5 symbols. So, the random variable the source generates takes 5 values, each value has some have some probability. Now, these are probabilities given: 0.2 0.05 0.45 0.15 and 0.15. Then which of these statements are true? There is a binary uniquely decodable code for this source with average length equal to 2. Now whether that is a source, there is a binary uniquely decodable code with length or not, that depends on that can be found out by taking the, by computing the entropy of this source.

(Refer Slide Time: 43:13)



Answer:

Entropy of the source

$$H(X) = 0.2\log(1/0.2) + 0.05\log(1/0.05) + 0.45\log(1/0.45) + 2\times 0.15\log(1/0.15) = 2.02$$

(a) False. Average length of any UDC $\geq H(X)$.
(b) True. There is a UDC with average length $\leq H(X)+1$.
(c) False. Check that the Huffman code has average length 2.5. There is no better code.

Dey Manjunath: Digital Communication - p.101/161

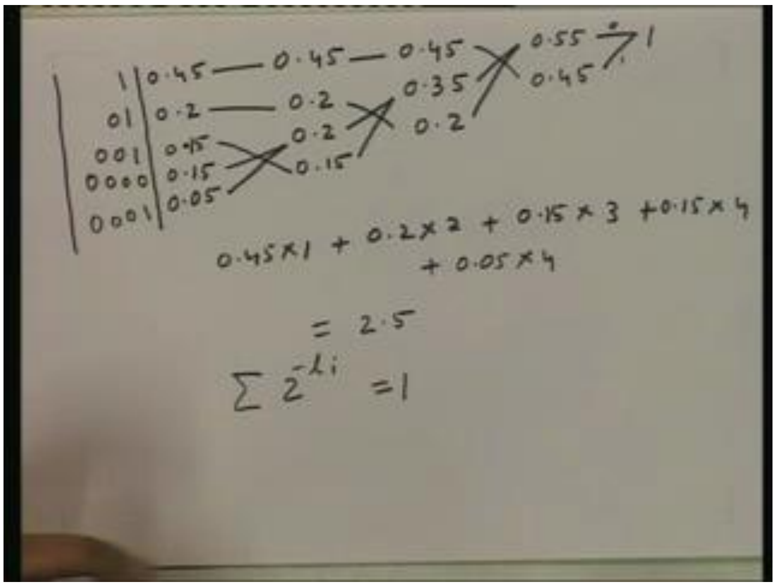
So, we need to compute the entropy of the source. We have it here; the entropy of the source can be this, the formula is simple p x times log 1 by p x, summation of all such terms. So, we take 0.2 probability 0.2 0.05 0.4 5 0.15, but that is 2 times; so 2 times that. And when we compute this we get 2.02. So, HX; the entropy of the source is 0.02. So, this statement, that there is a uniquely decodable code with average length 2 is false because, the entropy of the source itself is 2.02. So, we cannot have a code with average code length less than the entropy of the source. So, this statement is false.

Then, we have there is a binary uniquely decodable code for this source with average length less than 3.2. Now, what is the entropy of the source? 2.02. And we know that, there is always a code with average code length less than HX plus 1. So, HX is 2.02; so there is always a code with length 3.02. The entropy is 2.02; there will be code certainly with average code length less than 3.02. So, there this is true that, there is a uniquely decodable code with average length less than 3.2.

The next statement is, there is no binary uniquely decodable code for this source, with average length less than 2.5. Now, can you conclude about the truthness of this from the entropy? Entropy is 2.02. This number is 2.5, this is between 2.02 and 3.02; that means, this may possibly be average length of a code for the random variable because, we know that the optimal code length is between entropy and entropy plus 1. This number is such a number; it is between entropy and entropy plus 1.

So, this can be possibly a code length for a uniquely decodable code, for this random variable, but we, so this needs to be verified. We cannot grantee that, this will be certainly a code because this is between this, but the optimal code may not have this length. So, we need to check, we need to construct an optimal code and see what length it achieves. And we can do that, because Huffman code is optimal code and we know how to construct Huffman code.

(Refer Slide Time: 46:12)



$$0.45 \times 1 + 0.2 \times 2 + 0.15 \times 3 + 0.15 \times 4 + 0.05 \times 4$$

$$= 2.5$$

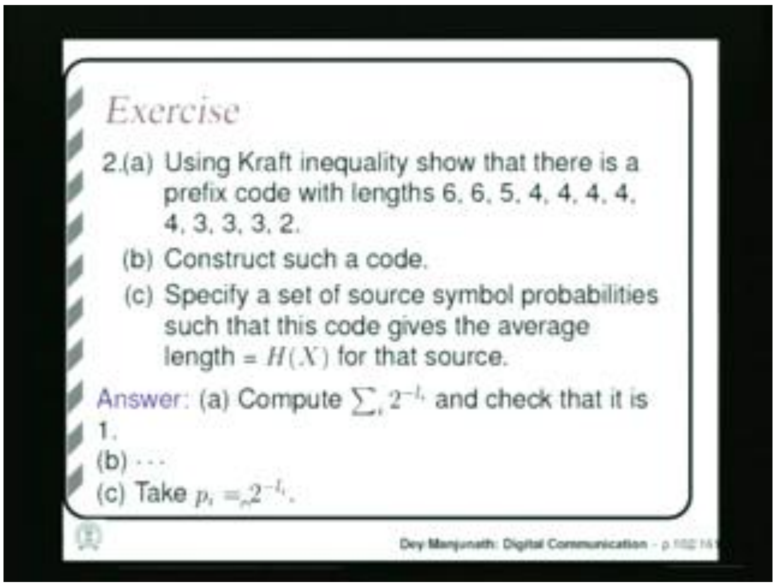
$$\sum 2^{-l_i} = 1$$

So, let us see, let us construct the Huffman code for this random variable and see if what average length we get. So, the code length the probabilities are 0.45 0.2 0.15 0.15 and then 0.05. So, first we combine this 2, we get 0.2 probability 0.2 0.45 0.2 0.2 then 0.15. So, we add 0 here and 1 here, then we combine we get 0.35. So, we add 0 to this; so, 0 to both this and 1 to this. Then we combine these 2 we get, 0.55 and this comes as it is. So, we add 0 here 0.35 0.35 comes from all these 3. And 0.45 and 0.2 gets 1 then, so we get, we add 0 to all this and 1 to here.

So, this the Huffman code for this random variable and what is the average code length? This is 0.45 times 1 plus 0.2 times 2 plus 0.15 times 3 plus 0.15 times 4 plus 0.05 times 4. When you compute this we will see that this is 2.5. Now, Huffman code itself has code length 2.5. So, certainly there is, there cannot be a better code. So, there is no binary uniquely decodable code for this source, with average length less than 2.5. So, this is, there is no binary uniquely decodable code for this source with average length less than

2.5 this is true, this is not correct There is no better code so this statement is true. There is no binary uniquely decodable code better than Huffman code.

(Refer Slide Time: 49:08)



Exercise

2.(a) Using Kraft inequality show that there is a prefix code with lengths 6, 6, 5, 4, 4, 4, 4, 4, 3, 3, 3, 2.

(b) Construct such a code.

(c) Specify a set of source symbol probabilities such that this code gives the average length = $H(X)$ for that source.

Answer: (a) Compute $\sum_i 2^{-l_i}$ and check that it is 1.

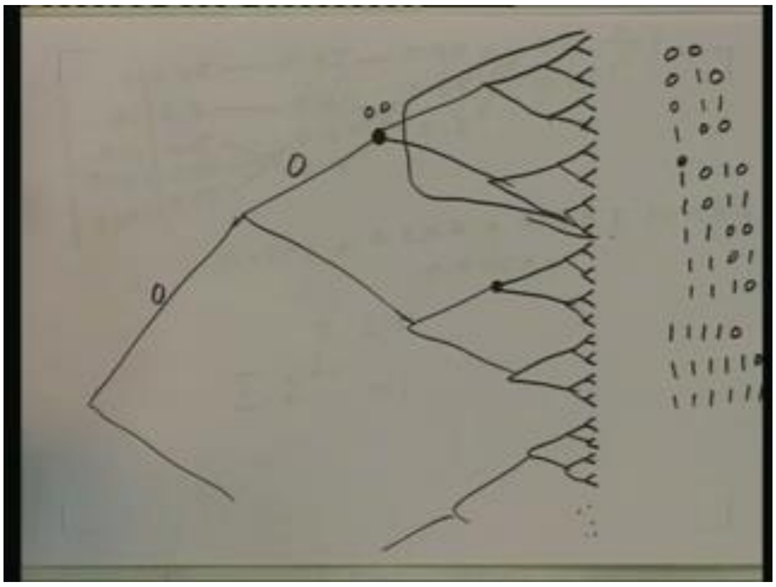
(b) $\cdots$

(c) Take $p_i = 2^{-l_i}$.

Dey Manjunath: Digital Communication - p.102/16

Next exercise is, using Kraft inequality show that there is a prefix code with lengths this. This can be checked because you simply compute, you simply compute this summation 2 power minus l i and l i is for li’s you can put these values 6 6 5 4 4 4 this. And then you will see that, this is actually equal to 1. So, this is, this satisfies Kraft inequality and construct such a code. Now, we have seen in the proof of Kraft inequality that, given certain lengths which satisfy Kraft inequality, a prefixed code can be constructed. The proof was by construction.

(Refer Slide Time: 50:04)



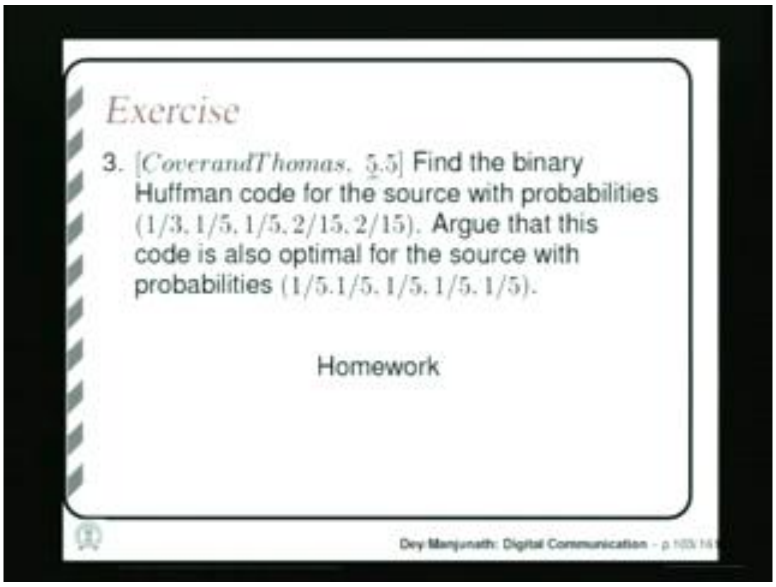
So, we can in fact, use that construction itself to construct such a code. So, let us do that, so we had, we have to the maximum code length is 6. So, we have to draw a code tree with 6 levels. So, 6 levels means: the number of, maximum number of branches at the last level will be 2 power 6 that is 64. So, we have, we will have last branches will come like this. So, 10 11 12 13 14 15 16 17 18 19 23 will be 32 such, there will 64 n here and then you can construct like this, you have to go on constructing, it will go on. There will be 64 such ends. And then we have to do this way. So, 1 2 3 4 5 1 2 3 4 5 6; 6 levels it will like this. From here, now you have to choose the code as we saw in the proof of that, proof of Kraft inequality that, we have to start from the smallest code length 2.

So, 2 code length mean: a code word of length 2 will be node at level 2 1 2. So, you take the first node at level 2. So, this is 1 code word. So, what is this code word? This is 00. So, we have 1 code word 0 0, then take a code of code word of length 3. So, we code word of length 3 at level 3, the first node available. Remember that this part is not available any more, this is disqualified by this. At level 3 we have this1 this is 010.

Similarly, as we go on taking we will see that was can get this 010011, then 100 these are 3 code words of length 3 then 4 5 code words of length 4. So, 1010 1011 1100 1101 1110. These are the 5 code words of length 4 then, 1 code word of length 5. So, we get 11110. This is 1 word length 5 then 2 code words of length 6. So, we have this code. So, as we take from the tree, also we will get this code. We have constructed such a code now, specify a set a symbol probabilities such that, this code gives the average length equal to HX. And we have seen that in the section with, when we discussed optimal codes we saw that, these are the probabilities to be taken.

So, you just take these lengths and compute these probabilities. If we compute these probabilities and take this code you can see that, this will achieve this code will achieve average code length equal to HX for this particular this particular source probabilities.

(Refer Slide Time: 54:04)



Exercise

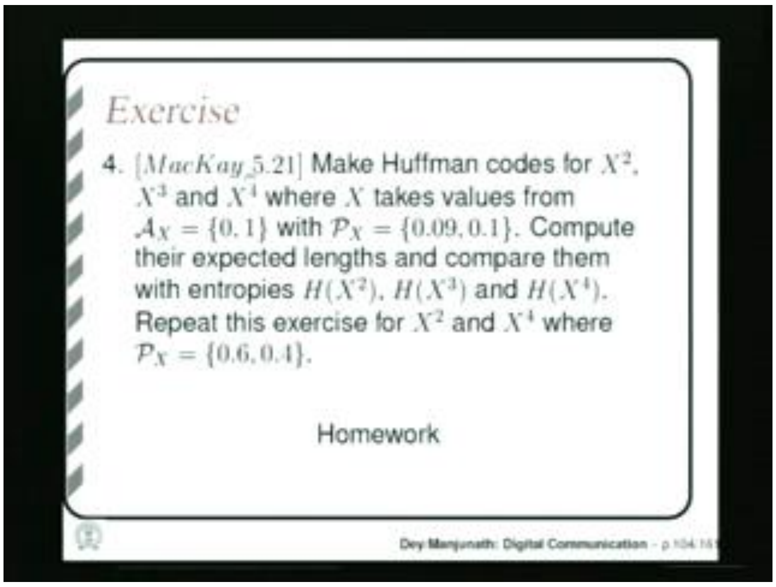
3. [$CoverandThomas,\ 5.5$] Find the binary Huffman code for the source with probabilities $(1/3, 1/5, 1/5, 2/15, 2/15)$. Argue that this code is also optimal for the source with probabilities $(1/5, 1/5, 1/5, 1/5, 1/5)$.

Homework

Dey Manjunath: Digital Communication - p.103/141

There are some other exercise, these will solve later in the later classes. This can be solved by this is given as homework also, you can note down this numbers this is from Thomas and Covers book 5.5 exercise number 5.5.

(Refer Slide Time: 54:25)



Exercise

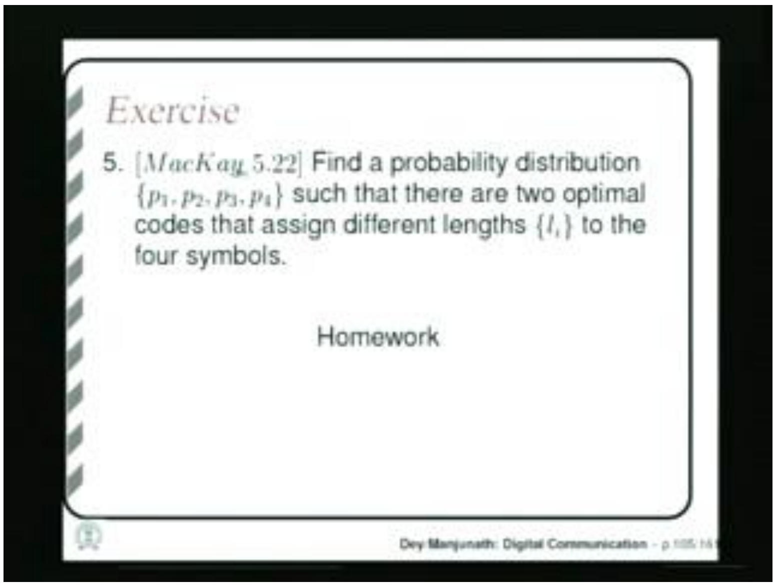
4. [$MacKay$ 5.21] Make Huffman codes for X^2, X^3 and X^4 where X takes values from $\mathcal{A}_X = \{0, 1\}$ with $\mathcal{P}_X = \{0.09, 0.1\}$. Compute their expected lengths and compare them with entropies $H(X^2)$, $H(X^3)$ and $H(X^4)$. Repeat this exercise for X^2 and X^4 where $\mathcal{P}_X = \{0.6, 0.4\}$.

Homework

Dey Manjunath: Digital Communication - p.104/161

This is from Mackays book exercise number 5.21.

(Refer Slide Time: 54:29)



Exercise

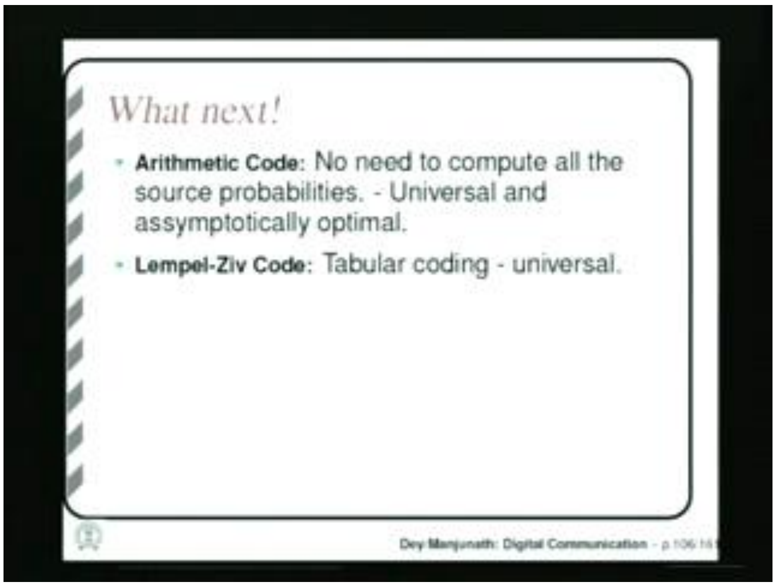
5. [$MacKay$ 5.22] Find a probability distribution $\{p_1, p_2, p_3, p_4\}$ such that there are two optimal codes that assign different lengths $\{l_i\}$ to the four symbols.

Homework

Dey Manjunath: Digital Communication - p.105/161

And this is from again Mackays book 5.22. Try to solve these problems as an exercise.

(Refer Slide Time: 54:36)



What next!

- **Arithmetic Code:** No need to compute all the source probabilities. - Universal and assymptotically optimal.
- **Lempel-Ziv Code:** Tabular coding - universal.

Dey Manjunath: Digital Communication - p.106/161

In the next class we will do arithmetic coding. We will see that using Shannon-Fano-Elias Codes principle itself, we can, we need not without computing source probabilities explicitly we can do arithmetic coding. And this code is universal works for any source probabilities and it is asymptotically normal because, it is basically Shannon-Fano-Elias Codes. So, it side asymptotically normal. And we will another source coding technique called Lempel-Ziv coding which does not require computation of probability at all. This is a basically tabular method.

So, we have done these 2 techniques. We have done these 2 topics in this class. We will do these 2 topics in the next classes. This is the end of this class. See you again in the next class.